IN THE CLAIMS:

Please cancel claims 21-23 and 25-30 without prejudice.

1. (Previously Presented) A computer-implemented method for parallel processing a sequence of requests received by an application server configured to execute a plurality of threads, comprising:

initiating, by the application server, a first thread configured to (i) perform the sequence of requests received from a user interacting with a web-based application and (ii) to maintain state information specific to the first thread, wherein each request, of the sequence, is composed as a uniform resource locater submitted to the web-based application;

initiating, by the application server, a second thread configured to perform the sequence of requests and to maintain state information specific to the second thread;

performing, sequentially, by the first thread, the sequence of requests;

performing, sequentially, by the second thread, the same sequence of requests, wherein the second thread performs requests, of the sequence, previously performed by the first thread a specified number of steps behind first thread and while the first thread continues to execute requests, from the sequence of requests, whereby the first thread and second thread each independently maintain state information regarding results of each request, of the sequence, performed by the first thread and the second thread, respectively;

upon completion of the sequence of requests by the first thread, producing a primary result for the sequence of requests;

subsequently, upon completion of the sequence of requests by the second thread, producing a secondary result for the sequence of requests;

displaying the primary result produced by the first thread; and

discarding, without displaying, the secondary result produced by the second thread.

2. (Previously Presented) The method of claim 1,

wherein the results of performing the sequence of requests by the first thread is visible to a user; and

wherein the results of performing the sequence of requests by the second thread is transparent to the user.

3. (Currently Amended) The method of claim 1, further comprising:

upon detecting an error in the results from performing, by the first thread, one of the sequence of requests, terminating the first thread; and

performing, by the second thread, [[,]] at least a portion of the sequence of requests performed by the terminated first thread and not yet performed by the second thread.

4-5. (Cancelled)

6. (Currently Amended) The method of claim 1, further comprising,

upon detecting an error in the results from performing, by the first thread, one of the sequence of [[requests;]] requests:

terminating execution of the first thread; and

returning to the user an indication of the request being performed by the first thread resulting in the detected error.

7. (Cancelled)

8. (Previously Presented) A computer-implemented method for parallel processing a sequence of requests received by an application server configured to execute a plurality of threads, comprising:

initiating, by the application server, a first thread configured to (i) perform the sequence of requests received from a user interacting with a web-based application and

(ii) to maintain state information specific to the first thread, wherein each request, of the sequence, is composed as a uniform resource locater submitted to the web-based application;

initiating, by the application server, a second thread configured to perform the sequence of requests and to maintain state information specific to the second thread;

performing, sequentially, by the first thread, the sequence of requests;

performing, sequentially, by the second thread, the same sequence of requests, wherein the second thread performs requests, of the sequence, previously performed by the first thread a specified number of steps behind first thread and while the first thread continues to execute requests, from the sequence of requests, whereby the first thread and second thread each independently maintain state information regarding results of each request, of the sequence, performed by the first thread and the second thread, respectively;

upon detecting an error in the results from performing, by the first thread, one of the sequence of requests:

terminating the first thread;

recovering from the error by executing remaining requests of the sequence by the second thread up to, but not including, the request resulting in the detected error;

prompting the user to modify the request which resulted in the detected error;

receiving a modification to the request;

executing the modified request;

executing any remaining requests, of the sequence of requests; and

displaying a final result produced by the second thread from executing (i) the sequence of requests up to, but not including, the request resulting in the detected error, (ii) the modified request and (iii) the remaining requests of the sequence, if any.

9. (Previously Presented) The method of claim 1, wherein the requests of the sequence are time ordered and processed by each of the first thread and the second thread according to the time order.

10. (Cancelled)

11. (Previously Presented) A computer-implemented method for parallel processing of requests received by an application server configured to execute a plurality of threads, comprising:

receiving a sequence of requests from a user, wherein each request, of the sequence, is composed as a uniform resource locater submitted to the web-based application;

placing the sequence of requests on a queue managed by the application server in a time-ordered manner;

performing, by a first thread managed by the application server, each request, of the sequence, upon being placed on the queue;

performing, by a second thread managed by the application server, at least a portion of the user requests on the queue step-wise with the first thread and N-requests behind the first thread; wherein the first thread and second thread each independently maintain state information regarding results of each request, of the sequence, performed by the first thread and the second thread, respectively;

upon completion of the sequence of requests by the first thread, producing a primary result for the sequence of requests;

subsequently, upon completion of the sequence of requests by the second thread, producing a secondary result for the sequence of requests;

displaying the primary result produced by the first thread; and

discarding, without displaying, the secondary result produced by the second thread.

12. (Currently Amended) The method of claim 11, further comprising:

upon detecting an error in the results from performing, by the first thread, one of the sequence of [[requests;]] requests:

terminating execution of the first thread; and

returning to the user an indication of the request performed by the first thread which resulted in the detected error.

13 - 42. (Cancelled)